Problem 1. Following is a recursive equation which obtained by dynamic approach.

$$S[i,w] = \begin{cases} -\infty & w < 0 \\ 0 & i = 0 \vee w = 0 \\ max(S[i-1,w], v_i + S[i-1, w-w_i]) & \text{otherwise} \end{cases}$$

where $S[i,w]$ is a maximum value which is less than or equal to w using subset of 1 to ith item. The solution of this knapsack problem is $S[n,W]$.

The time complexity of this recursive equation is obviously $O(nW)$, since only constant time is required to compute $S[i,w]$ and only $O(nW)$ possible indexes for S.

Also, the correctness of this equation is trivial. Since there are only two possible choices for ith item (pick or not), simply choose the maximum for these two cases. If ith item is selected, then the solution for $S[i,w]$ should be sum of ith item value and $S[i, w-w_i]$, otherwise just $S[i-1,w]$. For edge cases, if $w < 0$, it is invalid so $S[i,w] = 0$. Also, if nothing selected ($i = 0$) or weight is zero, $S[i,w]$ is always 0.

To construct the optimal subset itself, simply save the choices for ith item (pick or not) in other $O(nW)$ size array, and trace back from the final $S[n,W]$. Since saving these choices can be done in constant time, it does not affect to the time complexity.

Problem 2. Applying greedy approach to obtain the folowing algorithm.

procedure FKnapsack(n, W, v, w)
  $L := [(1, v_1, w_1), (2, v_2, w_2), \cdots, (n, v_n, w_n)]$
  $L := L.sort$(decreasing, using key as v_i/w_i)
  $R := [], V := 0$
  for $i = 1$ to n **do**
    $q := min(1, (W-V)/L[i].w)$
    $V := V + q \times L[i].v$
    $R := R.append((L[i].idx, q))$
  end for
  return R
end procedure

where R is a list of tuples (i, q) that denotes "select i th item as q"

The time complexity of this algorithm is $O(n \lg n)$, since sorting n values requires $O(n \lg n)$ times and for loop iterates $O(n)$, and its body can be done in constant time.

Now prove that this problem has greedy property.
(1) Assume that for every $i < j$, $v_i/w_i > v_j/w_j$.
(2) Let $X[i]$ is a fraction quantity of ith item for a solution X.
(3) Let R is a solution which obtained from the above algorithm.
(4) Suppose that there exist global optimal solution is R' which $R' \neq R$.
(5) Let k is the minimum index i that $R'[i] \neq R[i]$. Otherwise $R' = R$, so contradict to 4.

(6) Then, $R'[i] < R[i]$, since R is obtained by greedy choice that selecting highest density.
(7) Construct R'' from R' that add $R'[i] - R[i]$ amount to $R'[i]$ and subtract equal weight amount among $i + 1$ to nth item in R'.
(8) Then, R'' is equally optimal solution as R' does. Otherwise, contradict to 4.
(9) Simply repeat from step 5, k is increased at least 1, or k does not exist.
$\therefore$ Such global optimal solution R' which $R' \neq R$ does not exist, so R is global optimal.

Problem 3. Applying greedy approach to obtain the following algorithm

```
procedure TOMATO(d, n, C)
    B := [0, 0, ··· , 0] (size is n)
    T := [0, 0, ··· , 0] (size is n)
    front, end := 1, 1
    for i = 1 to n do
        while (front < end) ∧ (T[front] ≤ i − d) do
            front := front + 1
        end while
        while (front < end) ∧ (C[i] ≤ C[T[end − 1]]) do
            end := end − 1
        end while
        T[end] := i; end := end + 1
        B[T[front]] := B[T[front]] + 1
    end for
    return B
end procedure
```

The time complexity of this algorithm is $O(n)$, since the variable *end* can be increased at most n, so the two while loops body can be executed at most $O(n)$ during entire for loop.

The correctness of this algorithm is also trivial, since the tomato for day i should be bought in $[max(1, i-d+1), i]$. Therefore, optimum profilts can be obtained by simply find minimum price among d dates.